THE HONG KONG POLYTECHNIC UNIVERSITY
HONG KONG COMMUNITY COLLEGE

Subject Title	: Data Structures	**Subject Code**	: CCN2239
Session	: Semester One, 2018/19	**Time**	: 14:00 – 17:00
Date	: 10 December 2018	**Time Allowed**	: 3 Hours
Subject Examiner(s)	: Dr Pat CHAN Dr Joseph SO Dr Simon WONG		

This question paper has a total of **TWENTY** pages (including this covering page).

Instructions to Candidates:

1. There are THREE sections in this paper.
   - Section A (30%) – Multiple-choice Questions. Answer ALL questions in this section on the multiple-choice answer sheet provided. Each question carries 1 mark. Choose the BEST option for each question.
   - Section B (50%) – Short Questions. Answer any FIVE out of the SIX questions in this section in the answer book provided. Each question carries 10 marks. If you answer more than five questions, only the first five attempted questions will be marked. Indicate in your answer book clearly which five questions you are attempting.
   - Section C (20%) – Long Questions. Answer any ONE out of the TWO questions in this section in the answer book provided. Each question carries 20 marks. If you answer more than one question, only the first attempted question will be marked. Indicate in your answer book clearly which one question you are attempting.
2. Candidates are NOT allowed to retain the multiple-choice answer sheet, the answer book and the examination question paper.
3. Show all your work clearly and neatly. Marks will be deducted for untidy work.
4. Reasonable steps should be shown.
5. All programming code must be written in Java programming language.

Authorised Materials:

	YES	NO
CALCULATOR	[ ]	[✓]
SPECIFICALLY PERMITTED ITEMS	[ ]	[✓]

DO NOT TURN OVER THE PAGE UNTIL YOU ARE TOLD TO DO SO

Section B (50%) – Short Questions

Answer any <u>FIVE</u> out of the SIX questions in this section in the answer book provided. Each question carries 10 marks. If you answer more than five questions, only the first five attempted questions will be marked. Indicate in your answer book clearly which five questions you are attempting.

<u>Question B1</u>

(a) What are the differences between a binary tree and a tree? (4 marks)

(b) Figure 5 shows a BST with the keys 8 at the root and 3 at its left child. Draw the BST after inserting the following keys in sequence into the BST in Figure 5:

4, 6, 7, 5, 13, 15

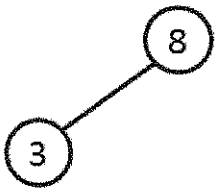


Figure 5

(6 marks)

<u>Question B2</u>

(a) What is an AVL tree? (2 marks)

(b) Draw the AVL tree after inserting the following keys in sequence:

98, 8, 26, 49, 5, 17, 63, 12

Show **<u>ALL</u>** the steps for each insertion. (8 marks)

<u>Question B3</u>

Implement the following static method in Java:

```
private static void merge(int[ ] data, int first,int n1, int n2);
```

Pre-condition: data has at least `n1+n2` components starting at `data[first]`. The first `n1` elements (from `data[first]` to `data[first+n1-1]`) are sorted in ascending order, and the last `n2` elements (from `data[first+n1]` to `data[first+n1+n2-1]`) are also sorted in ascending order.

Post condition: Starting at `data[first]`, `n1+n2` elements of data have been rearranged to be sorted in ascending order. (10 marks)

Question B4

(a) What is the difference between collision and overflow in *hashing*? (4 marks)

(b) Describe ANY **TWO** ways of solving collision problem in *hashing*? (4 marks)

(c) According to the answer in part (b), compare the complexity of inserting a dictionary pair (k_i, e_i) into a hash table **with** and **without** collision in terms of big-O. (2 marks)

Question B5

You are given the following Java `Passing` class.

```
public class Passing {
      public int x=2, y=5;

      public Passing() {
      }

      public Passing(int c, int d) {
            x=c;
            y=d;
            c=x*2;
            d=y*3;

      }
      public static int triArea(int a, int b ){
            return a++ * --b;
      }

      public static String[] changeEle(String[] gg) {
            String[] tempS = gg;
            String ts = gg[1];
            gg[1] = gg[0];
            gg[0] = ts;
            System.out.println();
            System.out.println("tempS[]: ");
            for (int i = 0; i<tempS.length;  i++) {
                  System.out.print( tempS[i] + "-");
            }

            return tempS;
      }

      public static void main(String[] args) {
            int a, b;
            int y = 0;

            a= 3;
            b= 9;

            System.out.println("The first product: " + triArea(a, b));
            Passing p = new Passing();
            System.out.println("1st x: " + p.x);
            System.out.println("1st y: " + p.y);
            System.out.println();
```

Question B5 (continued)

```
        int c= 5, d= 6;
        Passing q = new Passing(c,d);
        System.out.println("2nd c: " + c);
        System.out.println("2nd d: " + d);
        System.out.println("2nd x: " + q.x);
        System.out.println("2nd y: " + q.y);

        String[] s = {"MO", "EE", "DA", "SQ"};

        System.out.println();
        System.out.println("1st s[]: ");
        for (int i = s.length-1; i>0; i--) {
        System.out.print( s[i] + "-");
        }

        changeEle(s);
        System.out.println();
        System.out.println("2nd s[]: ");
        for (int i = 0; i<s.length;  i++) {
                    System.out.print(s[i] + "-");
        }

    }

}
```

Show the output when successfully executing *Passing* in Java. (10 marks)

Question B6

Given the following Java program code:

```
public class Furniture {
    private boolean replaceable;
    public static int cnt=0;
    protected String mainMaterial;
    public double weight;
    public int id;

    public Furniture () {
        mainMaterial = "wood";
        replaceable = true;
        cnt ++;
        id = cnt;
    }

    public void setReplaceable(boolean r) {
        replaceable = r;
    }

    public boolean getReplaceable() {
        return replaceable;
    }

    public double weightInPound() {
        return weight / 2.2;
    }
```

Question B6 (continued)

```
        public double transportationFee() {
                return weight * 500;
        }
}

public class Table extends Furniture {
        int width=3;
        public Table (int s) {
                width = s;
        }
        public void drawTopview() {
                String s= "";
                for (int i= 0; i<width; i++) {
                        s += "-";
                }
                for (int i=width; i>0; i--) {
                        System.out.println("+" +s+"+");
                }
        }


        public static void main(String[] args) {
                Table tA = new Table(5);
                tA.setReplaceable(false);
                tA.weight=55;
                System.out.println("TableA weight in pound: " +
                                                        tA.weightInPound());
                System.out.println("TableA transportation Fee: " +
                                                    tA.transportationFee());
                tA.drawTopview();
        }

}
```

(a) Show the output when successfully executing *Table* in Java. (6 marks)

(b) Add an instant method *picMe*() in Table.java so that the *id* is also displayed after executing *drawTopView*(). (4 marks)

- End of Section B -

Section C (20%) – Long Questions

Answer any <u>ONE</u> out of the TWO questions in this section in the answer book provided. Each question carries 20 marks. If you answer more than one question, only the first attempted question will be marked. Indicate in your answer book clearly which one question you are attempting.

<u>Question C1</u>

Given the following Java program codes

```
public class Dimension {
      public int height;
      public int length;
      public int width;

      public Dimension (int h, int len, int w) {
            height = h;
            length = len;
            width = w ;
      }

      public int objSize() {
            return height * length * width;
      }

      public int floorArea() {
            return length * width;
      }
}

public class Sofa{
      private String colour;
      private boolean isSoft;
      private int noOfSeats;
      private Dimension dim = new Dimension(0, 0, 0);
      private String shape = "circular";
      public static int cc=0;


      public Sofa() {
            this(3);
      }

      public Sofa(int se) {
            colour = "unknown";
            isSoft = true;
            noOfSeats = se;
            cc++;
      }

      public double objSize() {
            int cc = 5;
            return dim.objSize()*1.5;
      }

      public double floorSpace() {
            return dim.floorArea()*1.2;
      }
```

Question C1 (continued)

```
    public void setDimension (int h, int len,int w) {
        dim = new Dimension(h, len, w);
    }

    public void setSeatNo (int sn) {
        noOfSeats = sn;
    }

    public int getSeatNo () {
        return noOfSeats;
    }

    public String simPattern(int cnt, int vsh) {
        String s ="";
        int j =0;
        for (int i=0; i<cnt; i++) {
            j =0;
            while (j< i*10) {
                s += "^";
                j +=2;
            }
            s += "#\n";
        }
        return s;
    }

    public static void main(String[] avg) {
        Sofa comSo = new Sofa();
        String s = comSo.simPattern(4, 20);
        System.out.print(s);
        comSo.setDimension(10, 20, 15);
        System.out.println(comSo.objSize());

        Sofa hardSo = new Sofa();
        hardSo.setDimension(10, 20, 20);
        System.out.println(hardSo.floorSpace());

        System.out.println("Pieces of Furniture: " + Sofa.cc);

        Sofa lightSo = hardSo;
        lightSo.setDimension(5, 20, 30);
        System.out.println(lightSo.floorSpace());
        System.out.println(hardSo.floorSpace());
        System.out.println("\nNew pieces of Furniture: " + Sofa.cc);

    }
}
```

(a) Show the output when successfully executing *Sofa* in Java. (10 marks)

(b) Given the partially-completed Java `MovableSofa` class below:

Question C1 (continued)

```
public class MovableSofa extends Sofa {
      //
      // code to be completed in part (b)
      //

      public static void main(String[] args) {
            MovableSofa msFF = new MovableSofa(4);
            int wheelUnitCost = 5;
            System.out.println("The cost of wheels of msFF: " +
                                    msFF.wheelCost(wheelUnitCost));
            int noOfSeats = 3;
            MovableSofa msGG = new MovableSofa(6, noOfSeats);
            System.out.println("The number of seats of msGG: " +
                                    msGG.getSeatNo());
            int price = 1500; // the price of the msGG
            System.out.println("The price per seat of msGG: " +
                                    msGG.pricePerSeat(price));
      }
}
```

You are required to fill in the code so that it will give the following results after successful execution:

```
The cost of wheels of msFF: 20
The number of seats of msGG: 3
The price per seat of msGG: 500
```

Note: you may show only the added Java code in your answer. (10 marks)

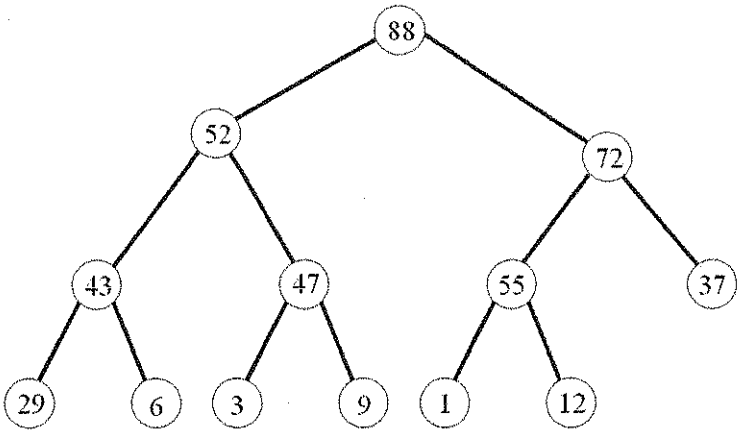
Question C2

Given the following Java Heap class:

```
public class Heap {
   int[] mheap;    // array for complete binary tree
   int size;       // number of elements in max heap

   public Heap(int[] mheap, int size) {
      this.mheap = mheap;
      this.size = 0;
   }

   public void put(int theKey) {
     //
     // code to be completed in part (a)
     //
   }

   public void heapify(int[] mheap, int c, int size) {
      int last = mheap[c];
      int child = c * 2 + 1;  // set left child
      while (child < size) {
         if (child + 1 < size && mheap[child] < mheap[child + 1])
            child++;
         if (last >= mheap[child])
            break;
         mheap[c] = mheap[child]; // move child up
         c = child; // move down a level
         child = c * 2 + 1;  // set left child
      }
      mheap[c] = last;
   }

   public int deleteMax(int[] mheap, int size) {
      int max = mheap[0]; // max element
      if (size > 0) {
         mheap[0] = mheap[--size];
         heapify(mheap, 0, size);
         this.size = size;
      }
      return max;
   }

   public void heapSort(int[] a, int size) { // code to be completed
   }

   public static void display(int[] a, int size) {
   // print elements in array
      for (int i = 0; i < size - 1; i++)
         System.out.print(a[i] + ", ");
      System.out.println(a[size - 1]);
   }
}
```

(a) Complete the `put()` method of the `Heap` class such that the `put()` method puts the key into the heap. (6 marks)

(b) Suppose the size of the heap `mheap` is large enough to accommodate a new key and the array representation of the heap `mheap` is shown in Figure 6. Show the result in both heap and its array representation after putting the key 69 into the heap.



Index →	0	1	2	3	4	5	6	7	8	9	10	11	12
	88	52	72	43	47	55	37	29	6	3	9	1	12

Figure 6

(4 marks)

(c) Based on the following Java `HeapTest` class:

```
public class HeapTest {
   public static void main(String [] args) {
      int[] a = {55, 42, 36, 69, 101, 13, 7, 28, -10};
      Heap h = new Heap(a, a.length);
      System.out.println("Elements in an array are:");
      h.display(a, a.length);
      for (int i = 0; i < a.length; i++)
         h.put(a[i]);
      System.out.println("The elements in the heap are:");
      h.display(h.mheap, h.size);
      System.out.println("After deleting the max,");
      h.deleteMax(h.mheap, h.size);
      System.out.println("the elements in the heap are:");
      h.display(h.mheap, h.size);
   }
}
```

show the output after successfully executing `java HeapTest`. (5 marks)

Question C2 (Continued)

(d) Based on the following Java `HeapSortTest` class:

```
public class HeapSortTest {
   public static void main(String [] args) {
      int[] a = {18, 22, 3, 65, 81, 100, 73, 21, 59};
      Heap h = new Heap(a, a.length);
      System.out.println("Elements in an array are:");
      h.display(a, a.length);
      h.heapSort(a, a.length);
      System.out.println("After sorting by heap sort, ");
      System.out.println("the elements in the array are:");
      h.display(a, a.length);;
   }
}
```

complete the `heapSort()` method of the `Heap` class such that the `heapSort()` method sorts the elements `a[0]` to `a[a.length - 1]` using the heap sort method and gives the following output after successfully executing `java HeapSortTest`.

```
Elements in an array are:
18, 22, 3, 65, 81, 100, 73, 21, 59
After sorting by heap sort,
the elements in the array are:
3, 18, 21, 22, 59, 65, 73, 81, 100
```

(5 marks)

- End of Section C -

- END OF PAPER -